

# OpenCMISS-iron examples and tests used by OpenCMISS developers at University of Stuttgart, Germany

Christian Bleiler\*, Andreas Hessenthaler\*,  
Thomas Klotz\*, Aaron Krämer<sup>†</sup>, Benjamin Maier<sup>‡</sup>,  
Sergio Morales\*, Mylena Mordhorst\*, Harry Saini\*

July 5, 2017  
10:51

## CONTENTS

1	Introduction	4
1.1	Cmgui files for cmgui-2.9	4
1.2	Variations to consider	4
1.3	Folder structure	4
2	How to work on this document	5
3	Until Friday	5
4	Immediately after Friday	5
5	Long-term	6
6	Diffusion equation	7
6.1	Equation in general form	7
6.2	Example-0001 [VALIDATED]	8
6.2.1	Mathematical model - 2D	8
6.2.2	Mathematical model - 3D	8
6.2.3	Computational model	8
6.2.4	Result summary	9
6.3	Example-0001-u [VALIDATED]	12
6.3.1	Mathematical model - 2D	12
6.3.2	Mathematical model - 3D	12
6.3.3	Computational model	12
6.3.4	Result summary	13
6.4	Example-0002 [VALIDATED]	16
6.4.1	Mathematical model - 2D	16
6.4.2	Mathematical model - 3D	16
6.4.3	Computational model	16
6.4.4	Result summary	17
6.5	Example-0003	20

\* Institute of Applied Mechanics (CE), University of Stuttgart, Pfaffenwaldring 7, 70569 Stuttgart, Germany

<sup>†</sup> Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany

<sup>‡</sup> Lehrstuhl Mathematische Methoden für komplexe Simulation der Naturwissenschaft und Technik, University of Stuttgart, Allmandring 5b, 70569 Stuttgart, Germany

6.5.1	Mathematical model - 2D	20
6.5.2	Mathematical model - 3D	20
6.5.3	Computational model	20
6.5.4	Result summary	21
6.6	Example-0004 [VALIDATED]	24
6.6.1	Mathematical model - 2D	24
6.6.2	Computational model	24
6.6.3	Result summary	24
6.7	Example-0011	26
6.7.1	Mathematical model - 2D	26
6.7.2	Mathematical model - 3D	26
6.7.3	Computational model	26
6.7.4	Result summary	27
7	Linear elasticity	30
7.1	Equation in general form	30
7.2	Example-0101 [PLAUSIBLE]	31
7.2.1	Mathematical model	31
7.2.2	Computational model	31
7.2.3	Results	32
7.2.4	Validation	32
7.3	Example-0102 [PLAUSIBLE]	35
7.3.1	Mathematical model	35
7.3.2	Computational model	35
7.3.3	Results	36
7.3.4	Validation	36
8	Finite elasticity	39
9	Navier-Stokes flow	40
9.1	Equation in general form	40
9.2	Example-0302-u	41
9.2.1	Mathematical model - 2D	41
9.2.2	Mathematical model - 3D	41
9.2.3	Computational model	41
9.2.4	Result summary	42
10	Monodomain	43
11	CellML model	44

## LIST OF FIGURES

Figure 1	2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].	9
Figure 2	2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].	10
Figure 3	3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].	10
Figure 4	3D results, current run w/ command line arguments [2.0 1.0 8 4 4 1 0].	11
Figure 5	2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].	13
Figure 6	2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].	14
Figure 7	3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].	14
Figure 8	3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].	15

Figure 9	2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0]. . . . .	17
Figure 10	2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0]. . . . .	18
Figure 11	3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0]. . . . .	18
Figure 12	3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0]. . . . .	19
Figure 13	2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0]. . . . .	21
Figure 14	2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0]. . . . .	22
Figure 15	3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0]. . . . .	22
Figure 16	3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0]. . . . .	23
Figure 17	2D results, iron reference w/ command line arguments [8 4 0 2 0]. . . . .	25
Figure 18	2D results, current run w/ command line arguments [8 4 0 2 0]. . . . .	25
Figure 19	2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1]. . . . .	27
Figure 20	2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1]. . . . .	28
Figure 21	3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1]. . . . .	28
Figure 22	3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0 1 1 1]. . . . .	29
Figure 23	Results, iron 2D fine mesh. . . . .	32
Figure 24	Results, iron 3D fine mesh. . . . .	32
Figure 25	Results, Abaqus 2D fine mesh. . . . .	33
Figure 26	Results, abaqus 3D fine mesh. . . . .	34
Figure 27	Results, iron 2D fine mesh. . . . .	36
Figure 28	Results, iron 3D fine mesh. . . . .	36
Figure 29	Results, Abaqus 2D fine mesh. . . . .	37
Figure 30	Results, abaqus 3D fine mesh. . . . .	38

## LIST OF TABLES

Table 1	Initials of people working on examples, in alphabetical order (surnames). . . . .	5
Table 2	Quantiative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extenions . . . . .	33
Table 3	Quantiative error between Abaqus 2017 and iron simulations for linear elastic shear . . . . .	37

## 1 INTRODUCTION

This document contains information about examples used for testing *OpenCMISS-iron*. Read: How-to<sup>1</sup> and [1].

### 1.1 Cmgui files for cmgui-2.9

### 1.2 Variations to consider

- Geometry and topology
  - 1D, 2D, 3D
  - Length, width, height
  - Number of elements
  - Interpolation order
  - Generated or user meshes
  - quad/hex or tri/tet meshes
- Initial conditions
- Load cases
  - Dirichlet BC
  - Neumann BC
  - Volume force
  - Mix of previous items
- Sources, sinks
- Time dependence
  - Static
  - Quasi-static
  - Dynamic
- Material laws
  - Linear
  - Nonlinear (Mooney-Rivlin, Neo-Hookean, Ogden, etc.)
  - Active (Stress, strain)
- Material parameters, anisotropy
- Solver
  - Direct
  - Iterative
- Test cases
  - Numerical reference data
  - Analytical solution
- A mix of previous items

### 1.3 Folder structure

TBD..

---

<sup>1</sup> <https://bitbucket.org/hessenthaler/opencmisshowto>

## 2 HOW TO WORK ON THIS DOCUMENT

In the Google Doc at [https://docs.google.com/spreadsheets/d/1RGKj8vVPqQ-PH0UwMX\\_e9TAzqaYavKi0z0D4pKY9RGI/edit#gid=0](https://docs.google.com/spreadsheets/d/1RGKj8vVPqQ-PH0UwMX_e9TAzqaYavKi0z0D4pKY9RGI/edit#gid=0) please indicate what you are working on or if a given example was finished

- no mark: to be done
- x: currently working on it
- xx: done

Initials	Full name
CB	Christian Bleiler
AH	Andreas Hessenthaler
TK	Thomas Klotz
AK	Aaron Krämer
BM	Benjamin Maier
SM	Sergio Morales
MM	Mylena Mordhorst
HS	Harry Saini

**Table 1:** Initials of people working on examples, in alphabetical order (surnames).

## 3 UNTIL FRIDAY

- Finish open examples
- Set a TAG for each example in document title:
  - DOCUMENTED: finish the documentation of the example (spatial domain, number of time steps, boundary conditions, etc.
  - COMPILES: example compiles (for default parameters)
  - RUNS: example runs (for default parameters)
  - CONVERGES: no convergence issues (for default parameters, results not plausible)
  - PLAUSIBLE: results look sensible (for default parameters)
  - VALIDATED: for all parameter sets it gives the correct results as compared to CHeart/Abaqus/analytical solution (includes visualisation scripts, run scripts, comparison scripts, documentation!, ...)
- Move progress Google-document into PDF-document
- Make from top directory - ensure all `run_example.sh` scripts are working as intended

## 4 IMMEDIATELY AFTER FRIDAY

- Move tags CONVERGE, PLAUSIBLE to VALIDATED
- Add GitHub issue for all tests/tags; VALIDATED means issue closed, else issue open
- Everybody runs everything!
- Meeting with Oliver
- Meeting with Auckland

## 5 LONG-TERM

- SMALL/BIG/PARALLEL targets
- Add more examples/those which were on the agenda but not started
- Jenkins
  - test SMALL/BIG/PARALLEL targets
  - integrate with GitHub (pull-requests triggers Jenkins, merge on success)

## 6 DIFFUSION EQUATION

### 6.1 Equation in general form

The governing equation is,

$$\partial_t \mathbf{u} + \nabla \cdot [\boldsymbol{\sigma} \nabla \mathbf{u}] = \mathbf{f}, \quad (1)$$

with conductivity tensor  $\boldsymbol{\sigma}$ . The conductivity tensor is,

- defined in material coordinates (fibre direction),
- diagonal,
- defined per element.

## 6.2 Example-0001 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

### 6.2.1 Mathematical model - 2D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1], \quad (2)$$

with boundary conditions

$$u = 0 \quad x = y = 0, \quad (3)$$

$$u = 1 \quad x = 2, y = 1. \quad (4)$$

No material parameters to specify.

### 6.2.2 Mathematical model - 3D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1] \times [0, 1], \quad (5)$$

with boundary conditions

$$u = 0 \quad x = y = z = 0, \quad (6)$$

$$u = 1 \quad x = 2, y = z = 1. \quad (7)$$

No material parameters to specify.

### 6.2.3 Computational model

- Commandline arguments are:

float: length along x-direction

float: length along y-direction

float: length along z-direction (set to zero for 2D)

integer: number of elements in x-direction

integer: number of elements in y-direction

integer: number of elements in z-direction (set to zero for 2D)

integer: interpolation order (1: linear; 2: quadratic)

integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

2.0 1.0 0.0 2 1 0 1 0

2.0 1.0 0.0 4 2 0 1 0

2.0 1.0 0.0 8 4 0 1 0

2.0 1.0 0.0 2 1 0 2 0

2.0 1.0 0.0 4 2 0 2 0

2.0 1.0 0.0 8 4 0 2 0

2.0 1.0 0.0 2 1 0 1 1

2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1



```

2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1

```

#### 6.2.4 Result summary

We use CHeart rev. 6292 to produce numerical reference solutions.

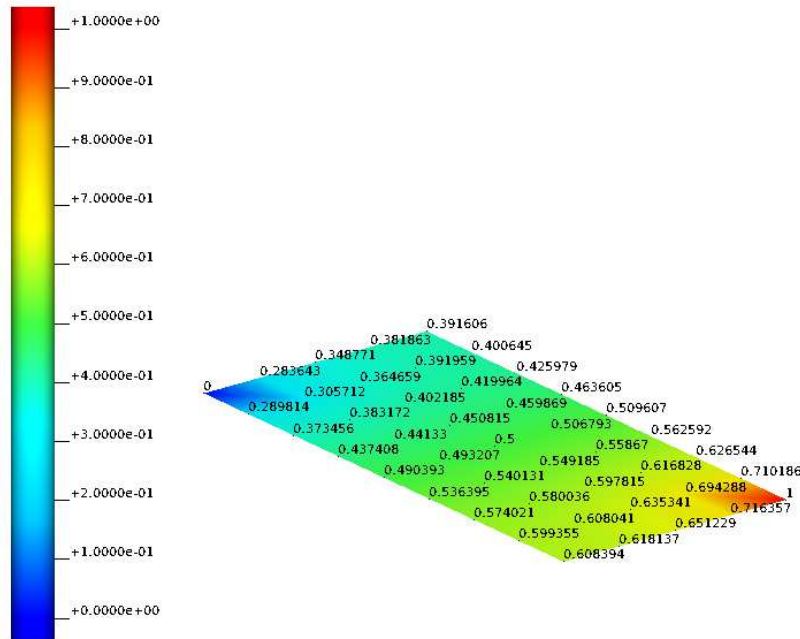


Figure 1: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

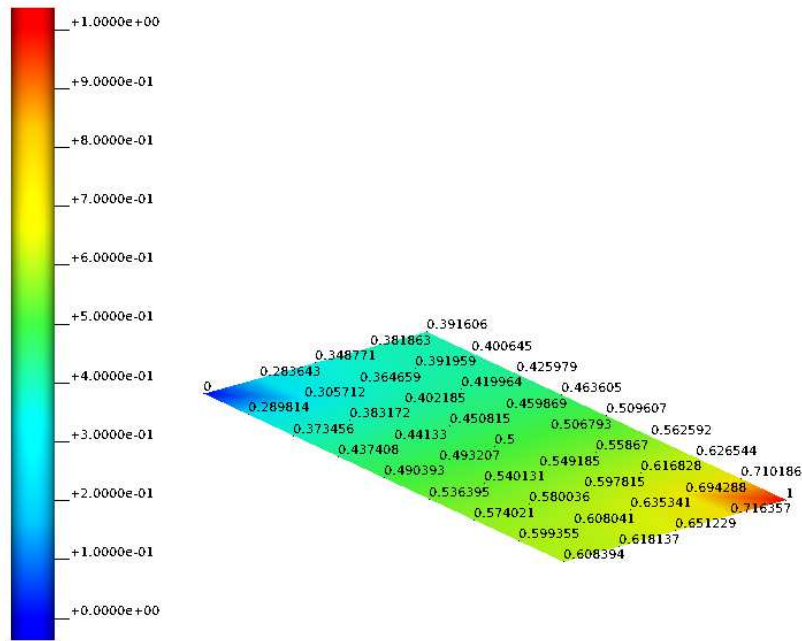


Figure 2: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

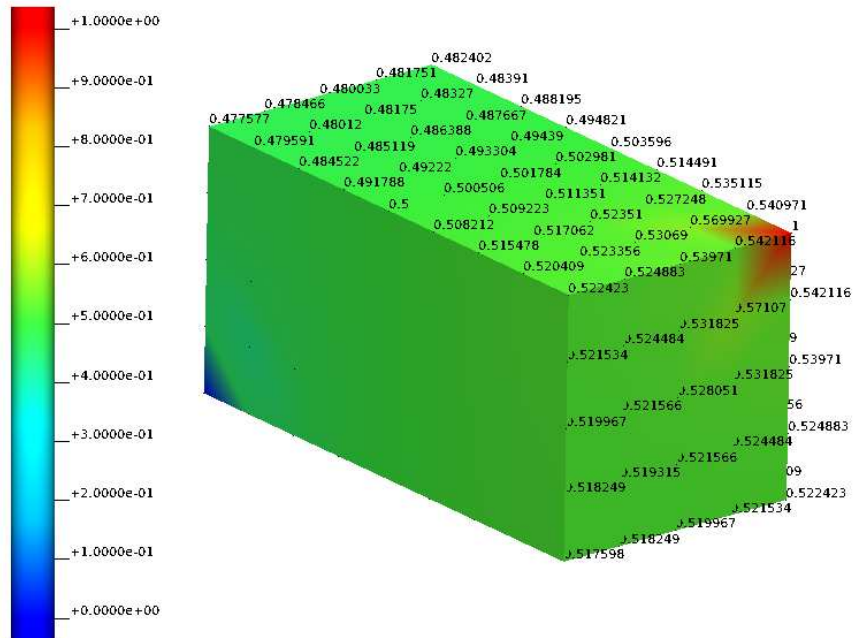


Figure 3: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 1 0].



### 6.3 Example-0001-u [VALIDATED]

Example uses user-defined regular meshes in CHeart mesh format and solves a static problem, i.e., applies the boundary conditions in one step.

#### 6.3.1 Mathematical model - 2D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1], \quad (8)$$

with boundary conditions

$$u = 0 \quad x = y = 0, \quad (9)$$

$$u = 1 \quad x = 2, y = 1. \quad (10)$$

No material parameters to specify.

#### 6.3.2 Mathematical model - 3D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1] \times [0, 1], \quad (11)$$

with boundary conditions

$$u = 0 \quad x = y = z = 0, \quad (12)$$

$$u = 1 \quad x = 2, y = z = 1. \quad (13)$$

No material parameters to specify.

#### 6.3.3 Computational model

- Commandline arguments are:

float: length along x-direction

float: length along y-direction

float: length along z-direction (set to zero for 2D)

integer: number of elements in x-direction

integer: number of elements in y-direction

integer: number of elements in z-direction (set to zero for 2D)

integer: interpolation order (1: linear; 2: quadratic)

integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

2.0 1.0 0.0 2 1 0 1 0

2.0 1.0 0.0 4 2 0 1 0

2.0 1.0 0.0 8 4 0 1 0

2.0 1.0 0.0 2 1 0 2 0

2.0 1.0 0.0 4 2 0 2 0

2.0 1.0 0.0 8 4 0 2 0

2.0 1.0 0.0 2 1 0 1 1

2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

```

2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1

```

- Note: Binary uses command line arguments to search for the relevant mesh files.

#### 6.3.4 Result summary

We use CHeart rev. 6292 to produce numerical reference solutions.

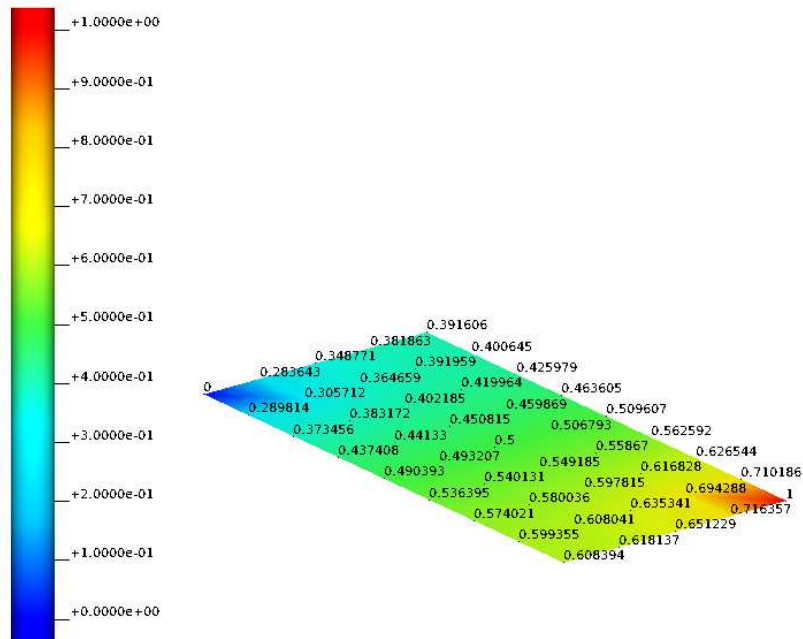


Figure 5: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

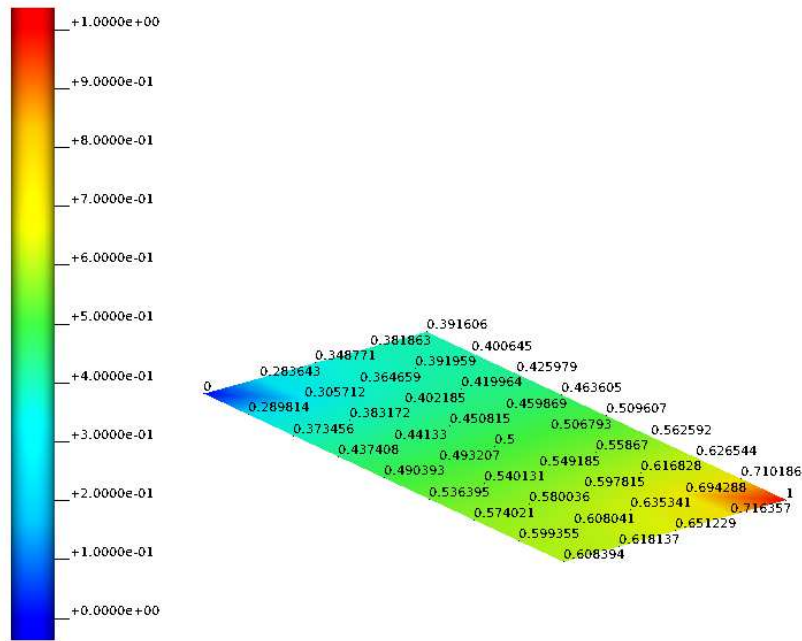


Figure 6: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

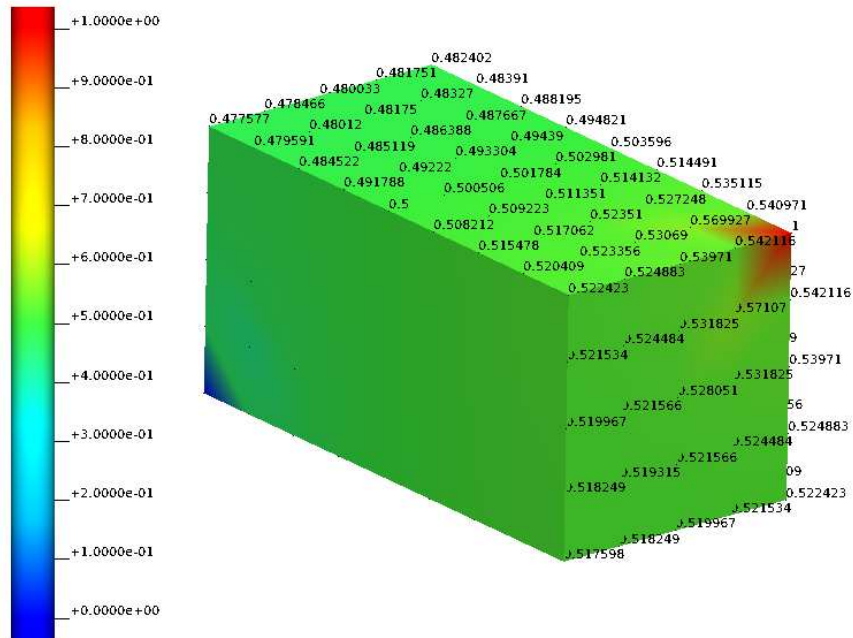


Figure 7: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 1 0].



## 6.4 Example-0002 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

### 6.4.1 Mathematical model - 2D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1], \quad (14)$$

with boundary conditions

$$u = 15y \quad x = 0, \quad (15)$$

$$u = 25 - 18y \quad x = 2. \quad (16)$$

No material parameters to specify.

### 6.4.2 Mathematical model - 3D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1] \times [0, 1], \quad (17)$$

with boundary conditions

$$u = 15y \quad x = 0, \quad (18)$$

$$u = 25 - 18y \quad x = 2. \quad (19)$$

No material parameters to specify.

### 6.4.3 Computational model

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
- Commandline arguments for tests are:
  - 2.0 1.0 0.0 2 1 0 1 0
  - 2.0 1.0 0.0 4 2 0 1 0
  - 2.0 1.0 0.0 8 4 0 1 0
  - 2.0 1.0 0.0 2 1 0 2 0
  - 2.0 1.0 0.0 4 2 0 2 0
  - 2.0 1.0 0.0 8 4 0 2 0
  - 2.0 1.0 0.0 2 1 0 1 1
  - 2.0 1.0 0.0 4 2 0 1 1
  - 2.0 1.0 0.0 8 4 0 1 1
  - 2.0 1.0 0.0 2 1 0 2 1



```

2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1

```

#### 6.4.4 Result summary

We use CHeart rev. 6292 to produce numerical reference solutions.

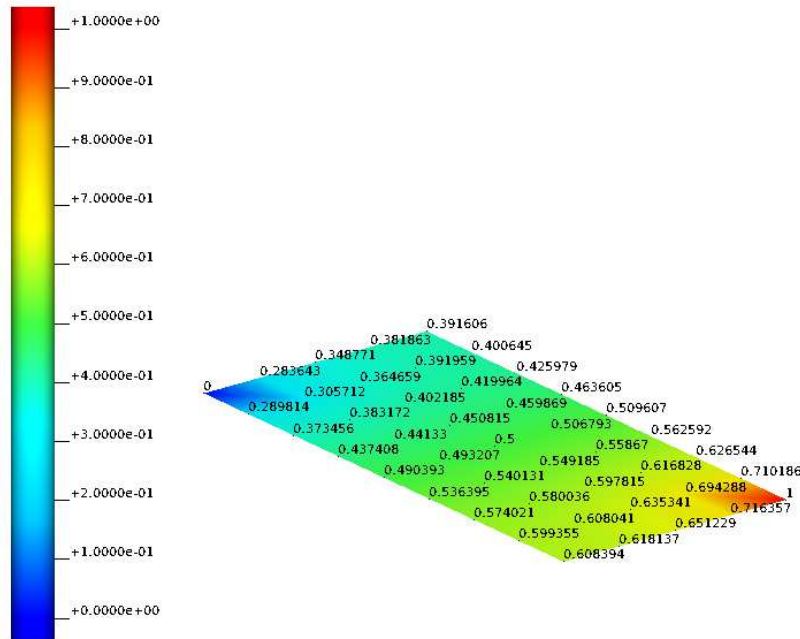


Figure 9: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

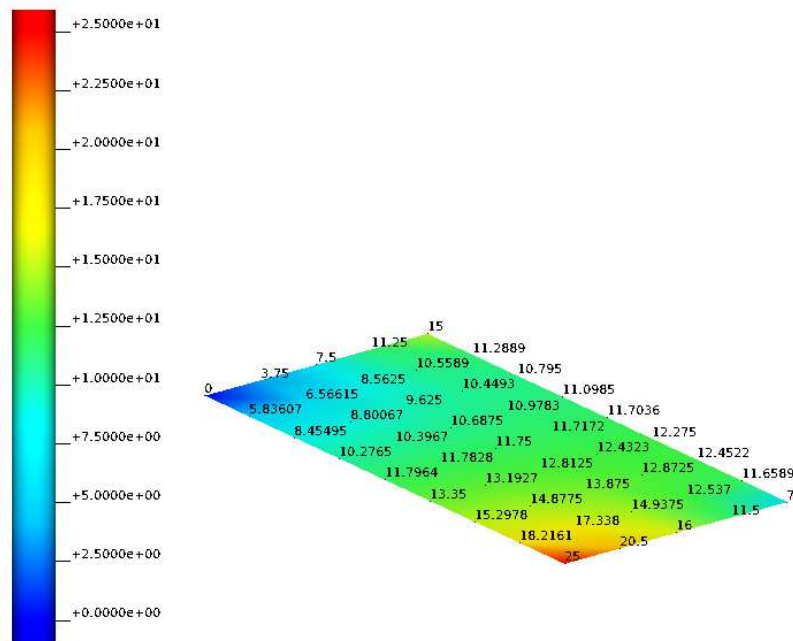


Figure 10: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

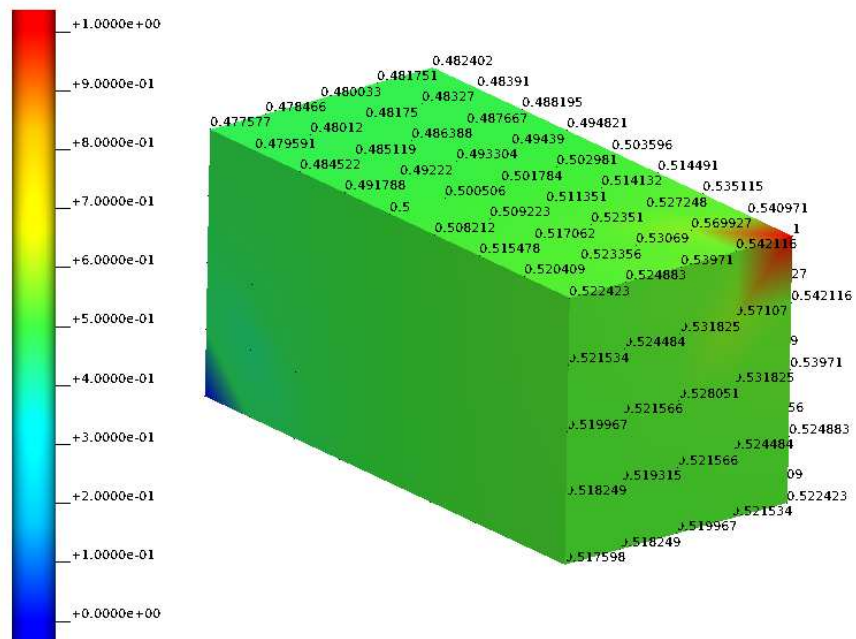


Figure 11: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

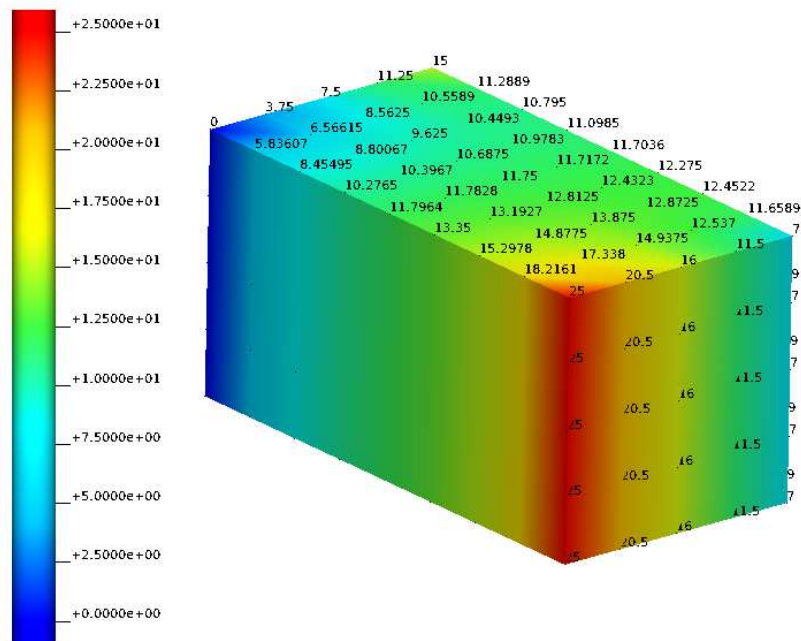


Figure 12: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

## 6.5 Example-0003

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

### 6.5.1 Mathematical model - 2D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1], \quad (20)$$

with boundary conditions

$$u = 15y \quad x = 0, \quad (21)$$

$$\partial_n u = 25 - 18y \quad x = 2. \quad (22)$$

No material parameters to specify.

### 6.5.2 Mathematical model - 3D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1] \times [0, 1], \quad (23)$$

with boundary conditions

$$u = 15y \quad x = 0, \quad (24)$$

$$\partial_n u = 25 - 18y \quad x = 2. \quad (25)$$

No material parameters to specify.

### 6.5.3 Computational model

- Commandline arguments are:

float: length along x-direction

float: length along y-direction

float: length along z-direction (set to zero for 2D)

integer: number of elements in x-direction

integer: number of elements in y-direction

integer: number of elements in z-direction (set to zero for 2D)

integer: interpolation order (1: linear; 2: quadratic)

integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

2.0 1.0 0.0 2 1 0 1 0

2.0 1.0 0.0 4 2 0 1 0

2.0 1.0 0.0 8 4 0 1 0

2.0 1.0 0.0 2 1 0 2 0

2.0 1.0 0.0 4 2 0 2 0

2.0 1.0 0.0 8 4 0 2 0

2.0 1.0 0.0 2 1 0 1 1

2.0 1.0 0.0 4 2 0 1 1

2.0 1.0 0.0 8 4 0 1 1

2.0 1.0 0.0 2 1 0 2 1

```

2.0 1.0 0.0 4 2 0 2 1
2.0 1.0 0.0 8 4 0 2 1
2.0 1.0 1.0 2 1 1 1 0
2.0 1.0 1.0 4 2 2 1 0
2.0 1.0 1.0 8 4 4 1 0
2.0 1.0 1.0 2 1 1 2 0
2.0 1.0 1.0 4 2 2 2 0
2.0 1.0 1.0 8 4 4 2 0
2.0 1.0 1.0 2 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1
2.0 1.0 1.0 8 4 4 1 1
2.0 1.0 1.0 2 1 1 2 1
2.0 1.0 1.0 4 2 2 2 1
2.0 1.0 1.0 8 4 4 2 1

```

#### 6.5.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

Figure 13: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

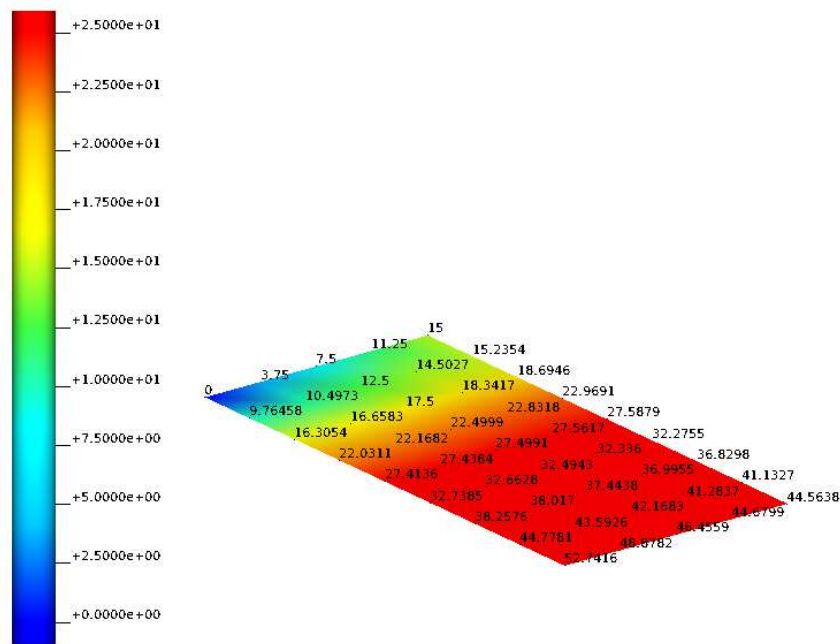


Figure 14: 2D results, current run w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0].

Figure 15: 3D results, iron reference w/ command line arguments [2.0 1.0 1.0 8 4 4 1 0].

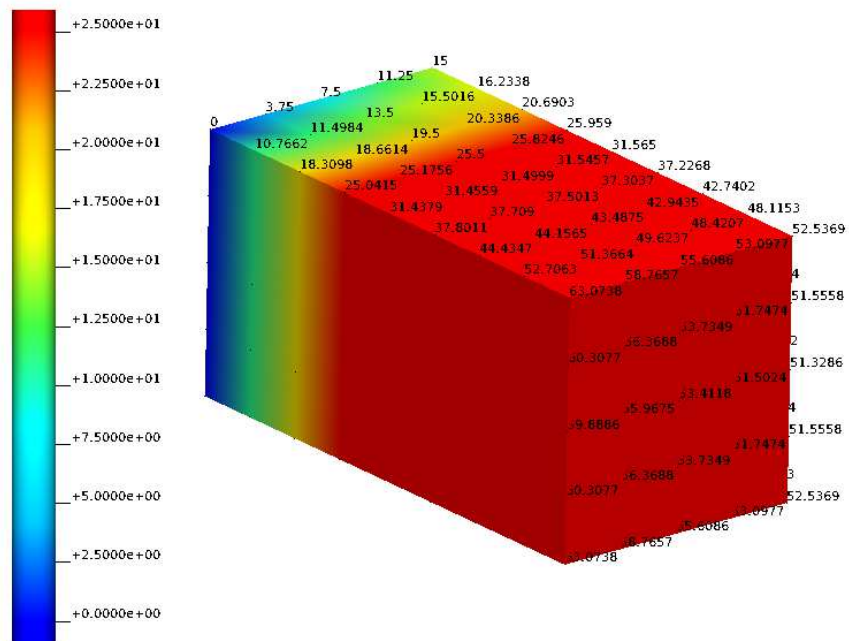


Figure 16: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 1 0].

## 6.6 Example-0004 [VALIDATED]

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

### 6.6.1 Mathematical model - 2D

We solve the following scalar equation,

$$\nabla \cdot \nabla u = 0 \quad \Omega = [0, 2] \times [0, 1], \quad (26)$$

with boundary conditions

$$u = 2.0e^x \cdot \cos(y) \quad \text{on } \partial\Omega. \quad (27)$$

No material parameters to specify.

### 6.6.2 Computational model

- Commandline arguments are:
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
- Commandline arguments for tests are:
  - 4 2 0 1 0
  - 8 4 0 1 0
  - 2 1 0 2 0
  - 4 2 0 2 0
  - 8 4 0 2 0
  - 4 2 0 1 1
  - 8 4 0 1 1
  - 2 1 0 2 1
  - 4 2 0 2 1
  - 8 4 0 2 1
  - 100 50 0 1 0 (not tested yet..)
  - 100 50 0 2 0 (not tested yet..)
  - 100 50 0 1 1 (not tested yet..)
  - 100 50 0 2 1 (not tested yet..)

### 6.6.3 Result summary

We use CHeart rev. 6292 to produce numerical reference solutions.



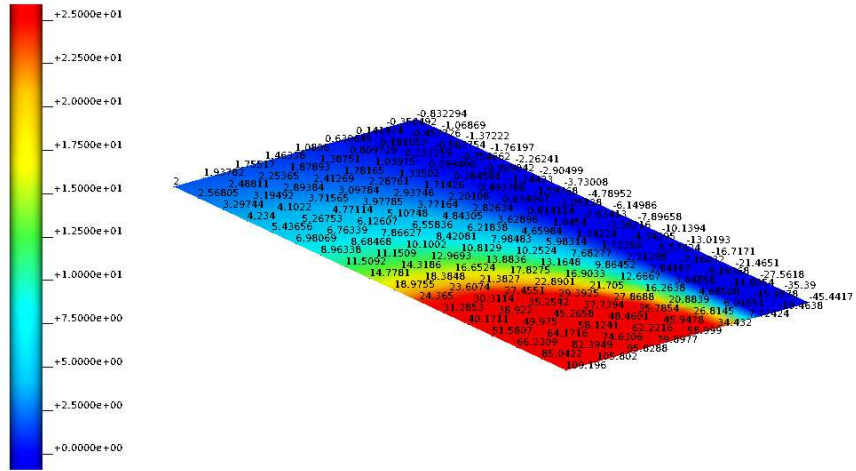


Figure 17: 2D results, iron reference w/ command line arguments [8 4 0 2 0].

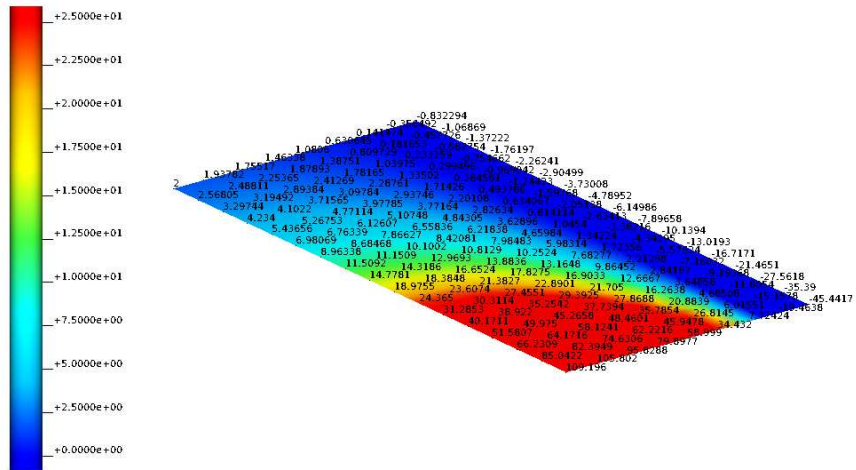


Figure 18: 2D results, current run w/ command line arguments [8 4 0 2 0].

## 6.7 Example-0011

Example uses generated regular meshes and solves a static problem, i.e., applies the boundary conditions in one step.

### 6.7.1 Mathematical model - 2D

We solve the following scalar equation,

$$\nabla \cdot [\sigma \nabla u] = 0 \quad \Omega = [0, 2] \times [0, 1], \quad (28)$$

with boundary conditions

$$u = 0 \quad x = y = 0, \quad (29)$$

$$u = 1 \quad x = 2, y = 1. \quad (30)$$

The conductivity tensor is defined as,

$$\sigma(x, t) = \sigma = \mathbf{I}. \quad (31)$$

### 6.7.2 Mathematical model - 3D

We solve the following scalar equation,

$$\nabla \cdot [\sigma \nabla u] = 0 \quad \Omega = [0, 2] \times [0, 1] \times [0, 1], \quad (32)$$

with boundary conditions

$$u = 0 \quad x = y = z = 0, \quad (33)$$

$$u = 1 \quad x = 2, y = z = 1. \quad (34)$$

The conductivity tensor is defined as,

$$\sigma(x, t) = \sigma = \mathbf{I}. \quad (35)$$

### 6.7.3 Computational model

- Commandline arguments are:

float: length along x-direction

float: length along y-direction

float: length along z-direction (set to zero for 2D)

integer: number of elements in x-direction

integer: number of elements in y-direction

integer: number of elements in z-direction (set to zero for 2D)

integer: interpolation order (1: linear; 2: quadratic)

integer: solver type (0: direct; 1: iterative)

float:  $\sigma_{11}$

float:  $\sigma_{22}$

float:  $\sigma_{33}$  (ignored for 2D)

- Commandline arguments for tests are:

2.0 1.0 0.0 2 1 0 1 0 1 1

2.0 1.0 0.0 4 2 0 1 0 1 1

2.0 1.0 0.0 8 4 0 1 0 1 1

2.0 1.0 0.0 2 1 0 2 0 1 1

```

2.0 1.0 0.0 4 2 0 2 0 1 1
2.0 1.0 0.0 8 4 0 2 0 1 1
2.0 1.0 0.0 2 1 0 1 1 1 1
2.0 1.0 0.0 4 2 0 1 1 1 1
2.0 1.0 0.0 8 4 0 1 1 1 1
2.0 1.0 0.0 2 1 0 2 1 1 1
2.0 1.0 0.0 4 2 0 2 1 1 1
2.0 1.0 0.0 8 4 0 2 1 1 1
2.0 1.0 1.0 2 1 1 1 0 1 1 1
2.0 1.0 1.0 4 2 2 1 0 1 1 1
2.0 1.0 1.0 8 4 4 1 0 1 1 1
2.0 1.0 1.0 2 1 1 2 0 1 1 1
2.0 1.0 1.0 4 2 2 2 0 1 1 1
2.0 1.0 1.0 8 4 4 2 0 1 1 1
2.0 1.0 1.0 2 1 1 1 1 1 1 1
2.0 1.0 1.0 4 2 2 1 1 1 1 1
2.0 1.0 1.0 8 4 4 1 1 1 1 1
2.0 1.0 1.0 2 1 1 2 1 1 1 1
2.0 1.0 1.0 4 2 2 2 1 1 1 1
2.0 1.0 1.0 8 4 4 2 1 1 1 1

```

#### 6.7.4 Result summary

We use CHeart rev. 6292 to produce numerical reference solutions.

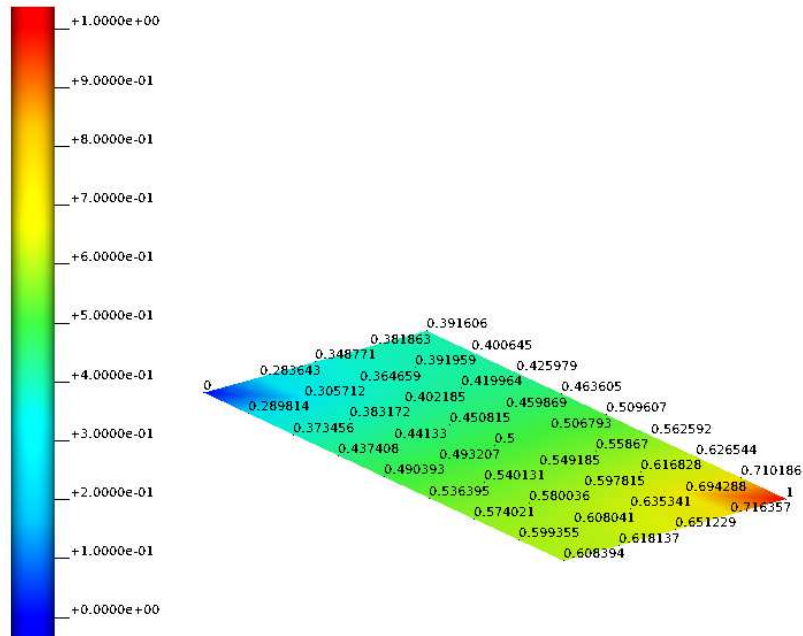


Figure 19: 2D results, iron reference w/ command line arguments [2.0 1.0 0.0 8 4 0 1 0 1 1].



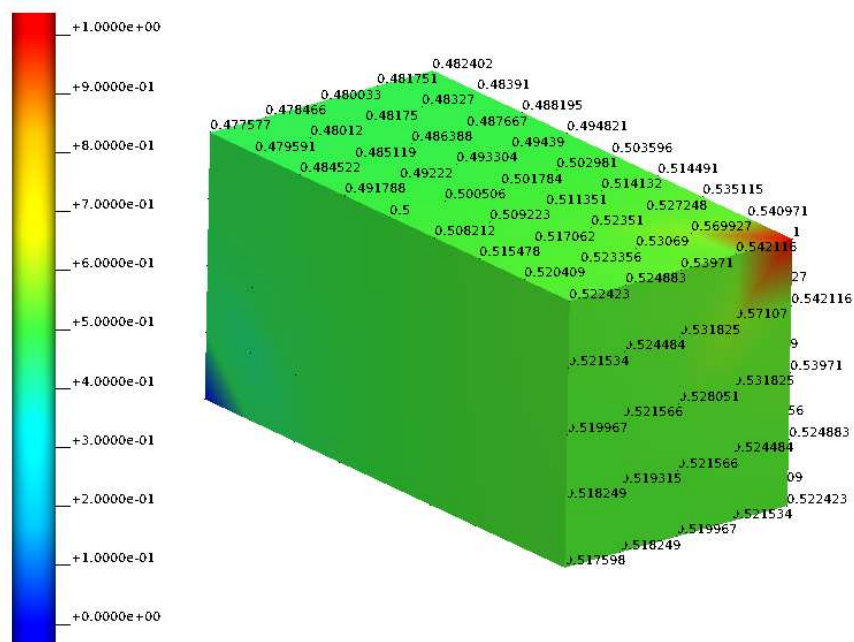


Figure 22: 3D results, current run w/ command line arguments [2.0 1.0 1.0 8 4 1 0 1 1 1].

## 7 LINEAR ELASTICITY

### 7.1 Equation in general form

$$\partial_{tt}\mathbf{u} + \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = \mathbf{f}(\mathbf{u}, t) \quad (36)$$

## 7.2 Example-0101 [PLAUSIBLE]

### 7.2.1 Mathematical model

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \sigma(\mathbf{u}, t) = 0 \quad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \quad (37)$$

with time step size  $\Delta_t = 1$  and  $\mathbf{u} = [u_x, u_y]$  in 2D  $\mathbf{u} = [u_x, u_y, u_z]$  in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \quad x = y = 0, \quad (38)$$

$$u_x = 16 \quad x = 160, \quad (39)$$

and in 3D by

$$u_x = u_y = u_z = 0 \quad x = y = z = 0, \quad (40)$$

$$u_x = 16 \quad x = 160. \quad (41)$$

The material parameters are

$$E = 10000 \text{MPa}, \quad (42)$$

$$\nu = 0.3, \quad (43)$$

$$\rho = 5 \times 10^{-9} \text{tonne} \cdot \text{mm}^3. \quad (44)$$

### 7.2.2 Computational model

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
  - float: elastic modulus
  - float: Poisson ratio
  - float: displacement percentage load
- Command line arguments for tests are:
  - 160 120 0 8 6 0 1 0 10000 0.3 0.05
  - 160 120 0 16 12 0 1 0 10000 0.3 0.05
  - 160 120 0 32 24 0 1 0 10000 0.3 0.05
  - 160 120 120 8 6 6 1 0 10000 0.3 0.05
  - 160 120 120 16 12 12 1 0 10000 0.3 0.05
  - 160 120 120 32 24 24 1 0 10000 0.3 0.05
  - 160 120 0 8 6 0 2 0 10000 0.3 0.05
  - 160 120 0 16 12 0 2 0 10000 0.3 0.05
  - 160 120 0 32 24 0 2 0 10000 0.3 0.05
  - 160 120 120 8 6 6 2 0 10000 0.3 0.05
  - 160 120 120 16 12 12 2 0 10000 0.3 0.05
  - 160 120 120 32 24 24 2 0 10000 0.3 0.05

### 7.2.3 Results

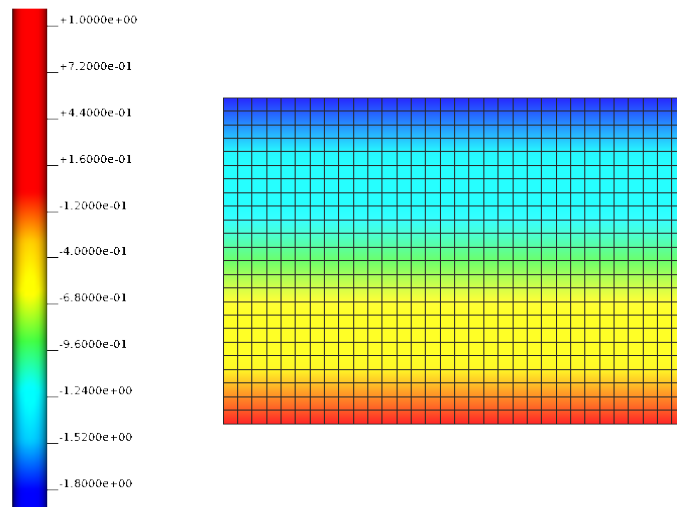


Figure 23: Results, iron 2D fine mesh.

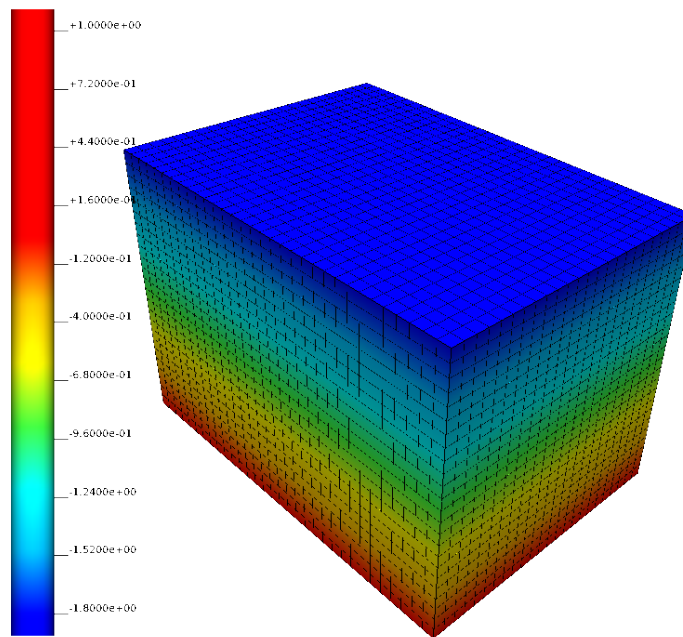


Figure 24: Results, iron 3D fine mesh.

### 7.2.4 Validation

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by



comparing the horizontal displacement  $u_y$  along the free-edge ( $y = 120$  for 2D and  $y = z = 120$  for 3D) and computing the L2-norm according to

$$L_2\text{-norm} = \frac{1}{N} \times \sum_{i=1}^N \sqrt{\left(u_{y,\text{abaqus}}^i - u_{y,\text{iron}}^i\right)^2}, \quad (45)$$

where  $N$  is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 2.

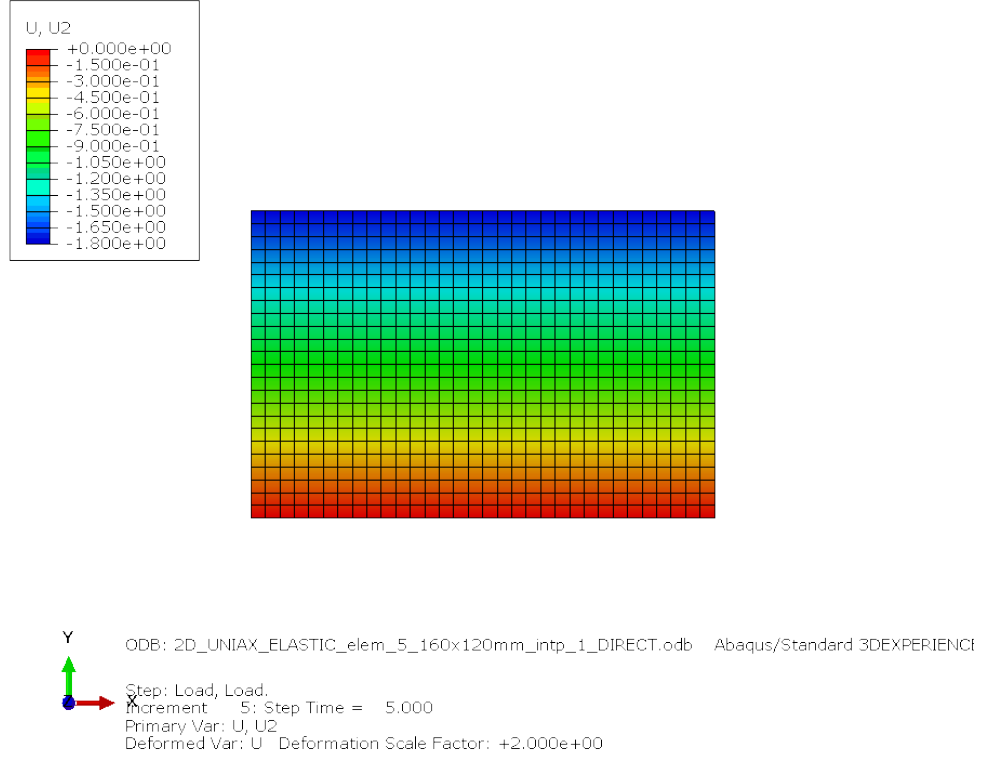


Figure 25: Results, Abaqus 2D fine mesh.

Dimension	Mesh	$L_2\text{-norm}$	Interpolation
2D	Coarse	$5.322 \times 10^{-16}$	Linear
2D	Medium	$1.559 \times 10^{-15}$	Linear
2D	Fine	$2.900 \times 10^{-15}$	Linear
3D	Coarse	$3.071 \times 10^{-17}$	Linear
3D	Medium	$2.125 \times 10^{-17}$	Linear
3D	Fine	$2.924 \times 10^{-17}$	Linear
2D	Coarse	$9.728 \times 10^{-16}$	Quadratic
2D	Medium	$2.039 \times 10^{-15}$	Quadratic
2D	Fine	$2.159 \times 10^{-15}$	Quadratic
3D	Coarse	$6.687 \times 10^{-16}$	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

Table 2: Quantitative error between Abaqus 2017 and iron simulations for linear elastic uniaxial extensions

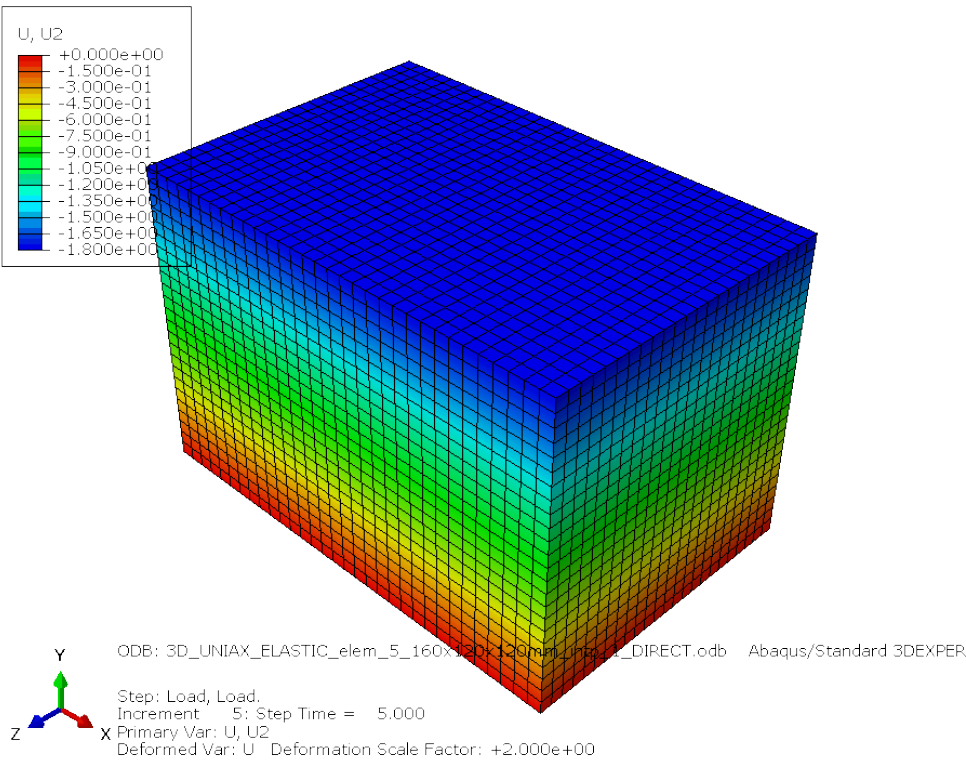


Figure 26: Results, abaqus 3D fine mesh.

### 7.3 Example-0102 [PLAUSIBLE]

#### 7.3.1 Mathematical model

We solve the following equation (both 2D and 3D domains are considered),

$$\nabla \cdot \boldsymbol{\sigma}(\mathbf{u}, t) = 0 \quad \Omega = [0, 160] \times [0, 120] \times [0, 120], t \in [0, 5], \quad (46)$$

with time step size  $\Delta_t = 1$  and  $\mathbf{u} = [u_x, u_y]$  in 2D  $\mathbf{u} = [u_x, u_y, u_z]$  in 3D. The boundary conditions in 2D are given by

$$u_x = u_y = 0 \quad y = 0, \quad (47)$$

$$u_y = 8 \quad x = 160, \quad (48)$$

and in 3D by

$$u_x = u_z = 0 \quad x = 0, \quad (49)$$

$$u_y = 0 \quad y = 0, \quad (50)$$

$$u_x = 160 \quad x = 160, \quad (51)$$

$$u_y = 8 \quad x = 160. \quad (52)$$

The material parameters are

$$E = 10000 \text{MPa}, \quad (53)$$

$$\nu = 0.3, \quad (54)$$

$$\rho = 5 \times 10^{-9} \text{tonne} \cdot \text{mm}^3. \quad (55)$$

#### 7.3.2 Computational model

- Commandline arguments are:
  - float: length along x-direction
  - float: length along y-direction
  - float: length along z-direction (set to zero for 2D)
  - integer: number of elements in x-direction
  - integer: number of elements in y-direction
  - integer: number of elements in z-direction (set to zero for 2D)
  - integer: interpolation order (1: linear; 2: quadratic)
  - integer: solver type (0: direct; 1: iterative)
  - float: elastic modulus
  - float: Poisson ratio
  - float: displacement percentage load
- Command line arguments for tests are:
  - 160 120 0 8 6 0 1 0 10000 0.3 0.05
  - 160 120 0 16 12 0 1 0 10000 0.3 0.05
  - 160 120 0 32 24 0 1 0 10000 0.3 0.05
  - 160 120 120 8 6 6 1 0 10000 0.3 0.05
  - 160 120 120 16 12 12 1 0 10000 0.3 0.05
  - 160 120 120 32 24 24 1 0 10000 0.3 0.05
  - 160 120 0 8 6 0 2 0 10000 0.3 0.05
  - 160 120 0 16 12 0 2 0 10000 0.3 0.05
  - 160 120 0 32 24 0 2 0 10000 0.3 0.05
  - 160 120 120 8 6 6 2 0 10000 0.3 0.05
  - 160 120 120 16 12 12 2 0 10000 0.3 0.05
  - 160 120 120 32 24 24 2 0 10000 0.3 0.05

### 7.3.3 Results

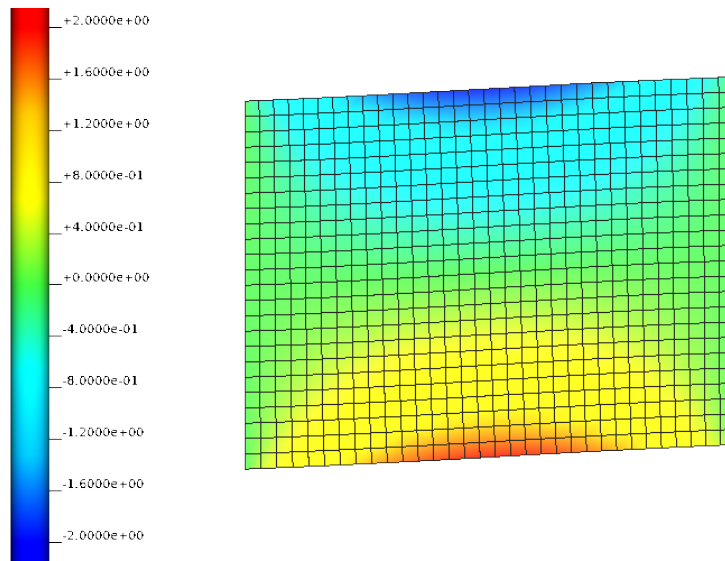


Figure 27: Results, iron 2D fine mesh.

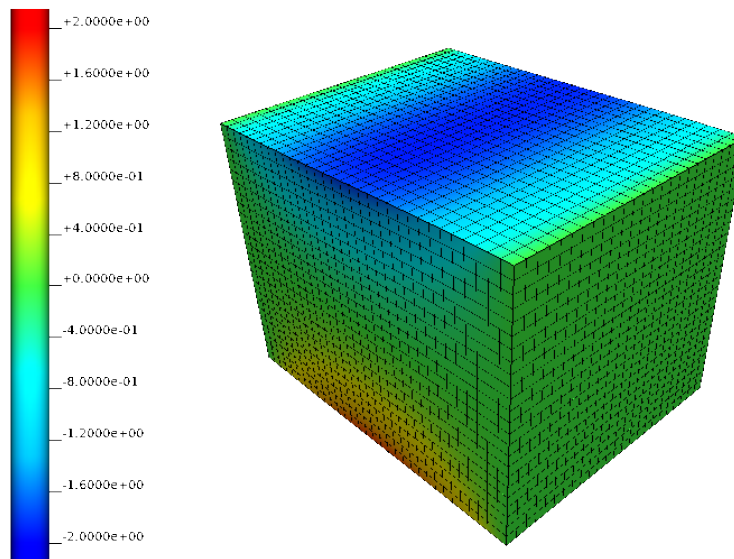


Figure 28: Results, iron 3D fine mesh.

### 7.3.4 Validation

The iron results are compared to those from Abaqus (version 2017). The figures below show selected results from the validation simulations carried out in Abaqus and provide a qualitative validation. A quantitative validation was carried out by

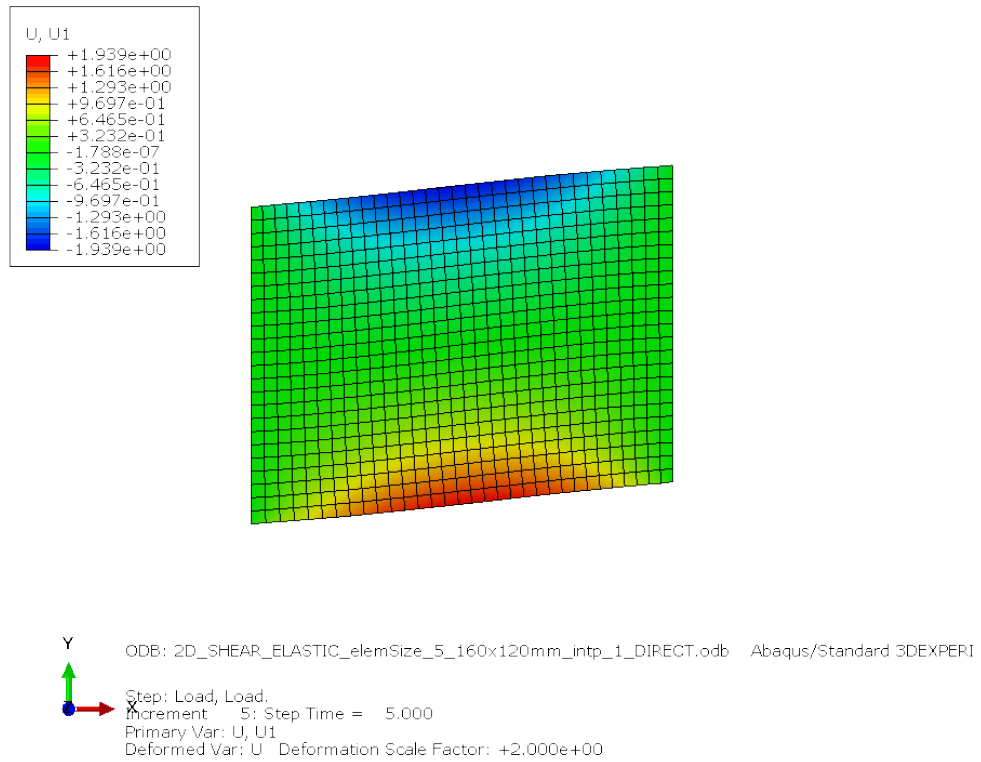
Dimension	Mesh	$L_2$ -norm	Interpolation
2D	Coarse	$6.696 \times 10^{-3}$	Linear
2D	Medium	$1.273 \times 10^{-3}$	Linear
2D	Fine	$2.489 \times 10^{-4}$	Linear
3D	Coarse	$4.234 \times 10^{-4}$	Linear
3D	Medium	$4.184 \times 10^{-5}$	Linear
3D	Fine	$3.781 \times 10^{-6}$	Linear
2D	Coarse	$3.036 \times 10^{-4}$	Quadratic
2D	Medium	$6.099 \times 10^{-5}$	Quadratic
2D	Fine	$1.089 \times 10^{-5}$	Quadratic
3D	Coarse	...	Quadratic
3D	Medium	...	Quadratic
3D	Fine	...	Quadratic

**Table 3:** Quantitative error between Abaqus 2017 and iron simulations for linear elastic shear

comparing the horizontal displacement  $u_x$  along the free-edge ( $y = 120$  for 2D and  $y = z = 120$  for 3D) and computing the  $L_2$ -norm according to

$$L_2\text{-norm} = \frac{1}{N} \times \sum_{i=1}^N \sqrt{\left(u_{y,\text{abaqus}}^i - u_{y,\text{iron}}^i\right)^2}, \quad (56)$$

where  $N$  is the total number of nodes along the free-edge. The results over the mesh refinements are given in Table 2.



**Figure 29:** Results, Abaqus 2D fine mesh.

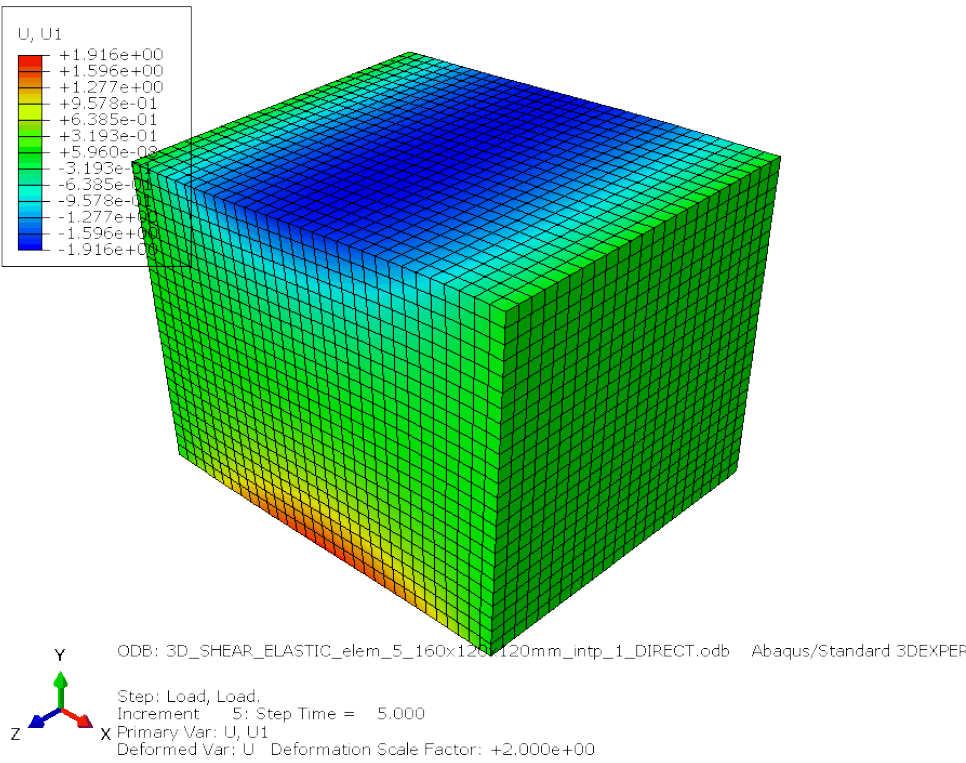


Figure 30: Results, abaqus 3D fine mesh.

## 8 FINITE ELASTICITY

## 9 NAVIER-STOKES FLOW

### 9.1 Equation in general form

$$\partial_t(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I}) = \rho \mathbf{f} \quad (57)$$



## 9.2 Example-0302-u

Example uses user-defined simplex meshes in CHeart mesh format with quadratic/linear interpolation for velocity/pressure and solves a dynamic problem.

Setup is the well-known lid-driven cavity problem on the unit square or unit cube in two and three dimensions

### 9.2.1 Mathematical model - 2D

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I}) = \rho \mathbf{f} \quad \Omega = [0, 1] \times [0, 1], \quad (58)$$

$$\nabla \cdot \mathbf{v} = 0, \quad (59)$$

with boundary conditions

$$\mathbf{v} = 0 \quad x = 0, \quad (60)$$

$$\mathbf{v} = 0 \quad x = 1, \quad (61)$$

$$\mathbf{v} = 0 \quad y = 0, \quad (62)$$

$$\mathbf{v} = [1, 0]^T \quad y = 1. \quad (63)$$

Density  $\rho = 1$ , viscosity  $\mu = 0.0025$ . Thus, Reynolds number  $Re = 400$ .

### 9.2.2 Mathematical model - 3D

We solve the incompressible Navier-Stokes equation,

$$\partial_t(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v} \otimes \mathbf{v} + p \mathbf{I}) = \rho \mathbf{f} \quad \Omega = [0, 1] \times [0, 1] \times [0, 1], \quad (64)$$

$$\nabla \cdot \mathbf{v} = 0, \quad (65)$$

with boundary conditions

$$\mathbf{v} = 0 \quad x = 0, \quad (66)$$

$$\mathbf{v} = 0 \quad x = 1, \quad (67)$$

$$\mathbf{v} = 0 \quad y = 0, \quad (68)$$

$$\mathbf{v} = [1, 0]^T \quad y = 1, \quad (69)$$

$$\mathbf{v} = 0 \quad z = 0, \quad (70)$$

$$\mathbf{v} = 0 \quad z = 1. \quad (71)$$

Density  $\rho = 1$ , viscosity  $\mu = 0.01$ . Thus, Reynolds number  $Re = 100$ .

### 9.2.3 Computational model

- Commandline arguments are:

integer: number of dimensions (2: 2D, 3: 3D)

integer: mesh refinement level (1, 2, 3, ...)

float: start time

float: stop time

float: time step size

float: density

float: viscosity

integer: solver type (0: direct; 1: iterative)

- Commandline arguments for tests are:

2 1 0.0 0.01 0.001 1.0 0.0025 0

- Note: Binary uses command line arguments to search for the relevant mesh files.

#### 9.2.4 *Result summary*

We use CHeart rev. 6292 to produce numerical reference solutions.

## 10 MONODOMAIN

## 11 CELLML MODEL

## REFERENCES

- [1] Chris Bradley, Andy Bowery, Randall Britten, Vincent Budelmann, Oscar Camara, Richard Christie, Andrew Cookson, Alejandro F Frangi, Thiranj Babarenda Gamage, Thomas Heidlauf, et al. Openmiss: a multi-physics & multi-scale computational infrastructure for the vph/physiome project. *Progress in biophysics and molecular biology*, 107(1):32–47, 2011.